DEFINITION

PROOF PROCEDURE

A *proof procedure* is a combination of an inference rule and an algorithm for applying that rule to a set of logical expressions to generate new sentences.

We present proof procedures for the *resolution* inference rule in Chapter 12.

DEFINITION

LOGICALLY FOLLOWS, SOUND, and COMPLETE

A predicate calculus expression X *logically follows* from a set S of predicate calculus expressions if every interpretation and variable assignment that satisfies S also satisfies X.

An inference rule is *sound* if every predicate calculus expression produced by the rule from a set S of predicate calculus expressions also logically follows from S.

An inference rule is *complete* if, given a set S of predicate calculus expressions, the rule can infer every expression that logically follows from S.

DEFINITION

MODUS PONENS, MODUS TOLLENS, AND ELIMINATION, AND INTRODUCTION, and UNIVERSAL INSTANTIATION

If the sentences P and $P \rightarrow Q$ are known to be true, then *modus ponens* lets us infer Q.

Under the inference rule *modus tollens*, if $P \rightarrow Q$ is known to be true and Q is known to be false, we can infer $\neg P$.

And elimination allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. For instance, $P \wedge Q$ lets us conclude P and Q are true.

And introduction lets us infer the truth of a conjunction from the truth of its conjuncts. For instance, if P and Q are true, then $P \wedge Q$ is true.

Universal instantiation states that if any universally quantified variable in a true sentence is replaced by any appropriate term from the domain, the result is a true sentence. Thus, if a is from the domain of X, $\forall X\ p(X)$ lets us infer p(a).

Resolution in predicate logic

- When are predicate instances the same?
  - is P(x) same as P(y)?
  - Can we resolve P(x) with $\neg P(5)$?
- **Substitution**: when a variable name is replaced by another variable or element of the domain.
  - Notation [x/a] means replacing all occurrences of x with a in the formula.
  - Example: substitution [x/5] in $P(x) \vee Q(x, y)$ results in $P(5) \vee Q(5, y)$
- **Unification:** matching literals and doing substitution so that resolution can be applied.

Resolution in predicate logic

- When are predicate instances the same?
  - If premises are $\forall x\ P(x)$ and $\forall y\ \neg P(y)$, can we resolve P(x) with $\neg P(y)$?
- **Substitution**: a variable name is replaced by another variable name or an element of the domain.
  - [Y/X]: change occurrences of X to Y.
    - [z/x]: $P(x) \vee Q(x,y)$ becomes $P(z) \vee Q(z,y)$
  - [a/X]: change occurrences of X to element a.
    - [5/x]: $P(x) \vee Q(x,y)$ becomes $P(5) \vee Q(5,y)$
- **Unification**: doing substitutions so that resolution could be applied.

Unification

- It is an algorithm for determining the substitutions needed to make two predicate logic expressions match.
- A variable cannot be unified with a term containing that variable. The test for it is called the occurs check.
  - e.g., cannot substitute X for X+Y in P$(X + Y)$
  - Most applicable when rather than having variables we have whole expressions (terms) evaluating to elements of the domain.
    - eg: x+y is a term: when $x, y \in \mathbb{Z}$, $x + y \in \mathbb{Z}$. With terms, can write formulas such as $P(x + y) \vee Q(y - 2)$

ALGORITHM TO CONVERT TO CLAUSAL FORM (1)

1. Eliminate conditionals →, using the equivalence

$$P \rightarrow Q = \neg P \vee Q$$

e.g, $(\exists X)\ (p(X) \wedge (\forall Y)\ (f(Y) \rightarrow h(X,Y)))$ becomes

$$(\exists X)\ (p(X) \wedge (\forall Y)\ (\neg f(Y) \vee h(X,Y)))$$

2. Eliminate negations or reduce the scope of negation to one atom.

e.g., $\neg \neg P = P$

$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

$$\neg (\exists X)\ p(X) = (\forall X) \neg p(X)$$

$$\neg (\forall X)\ p(X) = (\exists X) \neg p(X)$$

3. Standardize variables within a well-formed formula so that the bound or dummy variables of each quantifier have unique names.

e.g., $(\exists X) \neg p(X) \vee (\forall X)\ p(X)$ is replaced by

$$(\exists X) \neg p(X) \vee (\forall Y)\ p(Y)$$

ALGORITHM TO CONVERT TO CLAUSAL FORM (2)

4. ADVANCED STEP: if you have existential quantifiers, eliminate them by using Skolem functions, named after the Norwegian logician Thoralf Skolem.

e.g., $(\exists X)\ m(X)$ is replaced by $m(a)$

$(\forall X)\ (\exists Y)\ k(X, Y)$ is replaced by

$(\forall X)\ k(X, f(X))$

5. Convert the formula to prenex form which is a sequence of quantifiers followed by a matrix.

e.g., $(\exists X)\ (p(X) \wedge (\forall Y)\ (\neg f(Y) \vee h(X,Y)))$ becomes

$(\forall Y)\ (p(a) \wedge (\neg f(Y) \vee h(a,Y)))$

6. Convert the matrix to conjunctive normal form, which is a conjunctive of clauses. Each clause is a disjunction.

e.g., $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

7. Drop the universal quantifiers.

e.g., the formula above becomes $p(a) \wedge (\neg f(Y) \vee h(a,Y))$

ALGORITHM TO CONVERT TO CLAUSAL FORM (3)

8. Eliminate the conjunctive signs by writing the formula as a set of clauses

e.g., $p(a) \wedge (\neg f(Y) \vee h(a,Y))$ becomes $p(a), (\neg f(Y) \vee h(a,Y))$

9. Rename variables in clauses, if necessary, so that the same variable name is only used in one clause.

e.g., $p(X) \vee q(X) \vee k(X,Y)$ and $\neg p(X) \vee q(Y)$ become

$p(X) \vee q(X) \vee k(X,Y)$ and $\neg p(X1) \vee q(Y1)$

Anyone passing his history exams and winning the lottery is happy.

$\forall$ **X (pass (X,history) $\wedge$ win (X,lottery) $\rightarrow$ happy (X))**

Anyone who studies or is lucky can pass all his exams.

$\forall$ **X** $\forall$ **Y (study (X) $\vee$ lucky (X) $\rightarrow$ pass (X,Y))**

John did not study but he is lucky.

$\neg$ **study (john) $\wedge$ lucky (john)**

Anyone who is lucky wins the lottery.

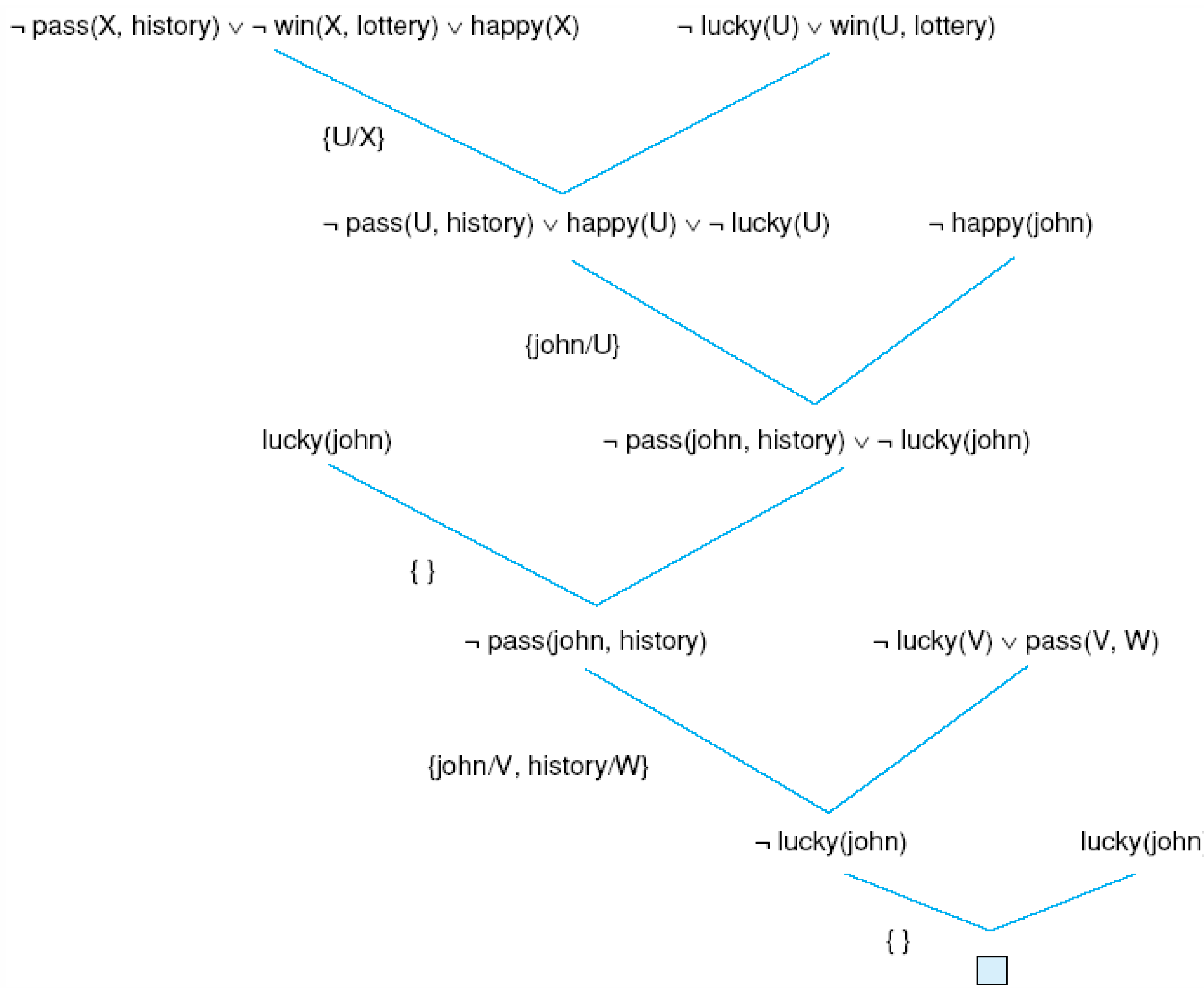
$\forall$ **X (lucky (X) $\rightarrow$ win (X,lottery))**

These four predicate statements are now changed to clause form (Section 12.2.2):

1. $\neg$ **pass (X, history) $\vee$ $\neg$ win (X, lottery) $\vee$ happy (X)**
2. $\neg$ **study (Y) $\vee$ pass (Y, Z)**
3. $\neg$ **lucky (W) $\vee$ pass (W, V)**
4. $\neg$ **study (john)**
5. **lucky (john)**
6. $\neg$ **lucky (U) $\vee$ win (U, lottery)**

Into these clauses is entered, in clause form, the negation of the conclusion:

7. $\neg$ **happy (john)**


¬ pass(X, history) ∨ ¬ win(X, lottery) ∨ happy(X)
¬ lucky(U) ∨ win(U, lottery)
{U/X}
¬ pass(U, history) ∨ happy(U) ∨ ¬ lucky(U)
¬ happy(john)
{john/U}
lucky(john)
¬ pass(john, history) ∨ ¬ lucky(john)
{ }
¬ pass(john, history)
¬ lucky(V) ∨ pass(V, W)
{john/V, history/W}
¬ lucky(john)
lucky(john)
{ }